Paralelización de Algoritmos de Seguimiento de Trayectorias

María Carina Roldan[1], Marcelo Naiouf [2], Armando De Giusti [3]
mcroldan@vittal.com.ar, {mnaiouf,degiusti}@lidi.info.unlp.edu.ar

LIDI. Laboratorio de Investigación y Desarrollo en Informática [4]
Facultad de Informática. UNLP

Abstract

En numerosas aplicaciones se debe seguir la trayectoria de una entidad, característica u objeto sobre una secuencia de imágenes. Cuando el procesamiento debe realizarse en tiempo real, trae aparejadas importantes restricciones y lleva a paralelizar los algoritmos de seguimiento.

El presente trabajo presenta los resultados de una implementación concreta. La misma trata el caso particular de objetos simples moviéndose en un contexto alcanzable por el elemento de visión (cámara de video). Se detallan las etapas involucradas en el desarrollo de la solución, especialmente en lo referente a su paralelización utilizando una red heterogénea de computadoras y soporte MPI (Message Passing Interface).

Finalmente, se realiza un análisis del comportamiento de los distintos algoritmos junto con la evaluación de los resultados obtenidos, la cual permite conocer la efectividad de la paralelización realizada, y determinar bajo qué condiciones resulta ser ésta la mejor solución.

Palabras Clave
Algoritmos Paralelos. Seguimiento de trayectorias. Visión por Computadora. Multiprocesadores heterogéneos. MPI.

Introducción

Existe un creciente interés en la automatización de procesos basados en la percepción visual del mundo que nos rodea, lo cual requiere la adquisición y procesamiento de imágenes. Esto tiene relación con el área que se ocupa de Visión por Computadora [Har92][Jai95]; tales sistemas tratan de imitar el proceso de visión humano en tareas con alto grado de dificultad en cuanto a exactitud y tiempo, o excesivamente rutinarias.

El objetivo de un sistema de visión por computadora es crear un modelo del mundo real a partir de imágenes, recuperando información útil acerca de una escena a partir de sus proyecciones 2D. Existen técnicas desarrolladas en otras áreas que son utilizadas para recuperar información desde imágenes. Algunas de ellas son el Procesamiento de Imágenes (generalmente en las primeras etapas para mejorar información particular y suprimir ruido) [Gon92][Bax94] [Jai89], Gráficos por Computadora (donde se generan imágenes a partir de primitivas geométricas), Reconocimiento de Patrones (el cual clasifica datos numéricos y simbólicos) [Sch92], etc.

Los sistemas de visión por computadora abarcan distintas aplicaciones, entre las cuales se pueden mencionar el diagnóstico de enfermedades mediante imágenes de tomografías computadas o el control de calidad de productos que van desde alimentos hasta piezas industriales [Law92]. Otra clase tiene que ver con la recuperación de la estructura tridimensional de un ambiente, es decir, *el reconocimiento e interpretación de los movimientos en una escena*; estas aplicaciones son útiles en la automatización de navegación de autos, aviones o robots. Un tercer grupo incluye el análisis y manejo de grandes volúmenes de datos obtenidos por satélites, utilizados para el pronóstico del tiempo, el estudio del comportamiento del planeta, la agricultura, etc.

---

[1] Estudiante Avanzada Licenciatura en Informática
[2] Profesor Adjunto Dedicación Exclusiva.
[3] Investigador Principal CONICET. Profesor Titular Dedicación Exclusiva.
[4] LIDI - Facultad de Informática. UNLP - 50 y 115 1er Piso, (1900) La Plata, Argentina. TE/Fax +(54)(221)422-7707. http://lidi.info.unlp.edu.ar

El seguimiento de los objetos en una secuencia de imágenes está particularmente asociado con:
- El reconocimiento y clasificación de objetos.
- El seguimiento de las trayectorias de los mismos a través de esa secuencia.
Además, se debe operar en tiempo real, lo que implica el uso de hardware especializado y algoritmos robustos para reducir la posibilidad de fallas, y los sistemas deben ser flexibles para adaptarse rápidamente a cambios en el proceso. La restricción temporal naturalmente lleva a pensar en una resolución del problema utilizando paralelismo para cumplir con los requerimientos [Akl97][Bri95][Tin98][Zom96].

En el presente trabajo se analizaron las etapas relacionadas con los sistemas de visión por computadora, como el mecanismo de tracking y el reconocimiento de objetos en la secuencia de imágenes obtenidas con una video cámara. Se desarrolló una primera solución secuencial, consistente en un conjunto de algoritmos que permiten extraer de la secuencia la información de los objetos y sus movimientos. En una segunda etapa, se paralelizaron los algoritmos tanto del proceso de reconocimiento como del de seguimiento de las trayectorias. Esto se implementó sobre una red heterogénea y con un soporte de programación MPI. Las soluciones halladas permitieron realizar una comparación y evaluación de la performance, cuyos resultados se exponen al final del artículo.

Seguimiento de Trayectorias en Visión por Computadora

El objetivo de un sistema de visión por computadora es crear un modelo del mundo real a partir de imágenes. Por este motivo un campo estrechamente relacionado es el del procesamiento digital de imágenes, que incluye tópicos tales como el realce, la compresión y la corrección de imágenes borrosas o fuera de foco. Los algoritmos de visión por computadora toman imágenes como entrada y producen salidas tales como la representación del contorno de un objeto encontrado en una escena.

Los algoritmos de procesamiento de imágenes son útiles en las primeras etapas de un sistema de visión, usualmente para enfatizar información y suprimir ruido. El objetivo del *realce* es acentuar características para un posterior análisis o despliegue en un dispositivo de salida. La *segmentación* identifica los componentes semánticamente significativos en una imagen y agrupa los puntos que pertenecen a las mismas; el procedimiento se simplifica si las imágenes son nítidas, por lo que habitualmente el realce constituye un paso previo importante. El resultado de la segmentación es un conjunto de regiones u objetos, que pueden ser identificados por una ubicación o un conjunto de características.

Un sistema de *reconocimiento* de objetos encuentra ítems del mundo real a partir de una imagen capturada utilizando modelos conocidos a priori. El reconocimiento está relacionado con la segmentación, e implica considerar la forma de representación de los objetos, los atributos importantes a detectar, la manera de comparar las características con las de los modelos, etc. La complejidad del reconocimiento depende de varios factores, como por ejemplo, si las condiciones en las que se capturó una imagen (iluminación, fondo, punto de vista) no son similares a aquellas bajo las cuales se definió el modelo. Otro factor es la oclusión: si hay sólo un objeto puede ser completamente visible, pero a medida que crece la cantidad, la posibilidad de oclusión aumenta y se dificulta el proceso de reconocimiento.

Visión Dinámica

Si bien los primeros sistemas de visión por computadora estaban relacionados principalmente con escenas estáticas, para varias aplicaciones se han diseñado soluciones que analizan escenas dinámicas. La entrada a los mismos es una secuencia de cuadros tomados de una escena real con movimientos. Cada cuadro representa una imagen de la escena en un instante de tiempo, los cambios en la misma pueden deberse al movimiento de la cámara, al movimiento de los objetos, a modificaciones en la iluminación o en la forma o tamaño de los objetos. El sistema debe detectar cambios, determinar las características de los movimientos, recuperar la estructura de los objetos y reconocer los mismos.

Al analizar una escena dinámica existen cuatro posibilidades: cámara estacionaria y objetos estacionarios (SCSO), cámara estacionaria y objetos en movimiento (SCMO), cámara en movimiento y objetos estacionarios (MCSO), cámara en movimiento y objetos en movimiento (MCMO). Según el caso, se requieren

distintas técnicas. SCMO ha recibido la mayor atención; usualmente el objetivo es detectar movimiento, reconocer los objetos que se mueven y computar las características de esos movimientos.

El proceso de *correspondencia* trata de identificar el mismo objeto en dos o más cuadros y determinar cambios en su ubicación (esto es, su movimiento). Se trata de aparear un punto $p_i = (x_i, y_i)$ de una imagen, con un punto $p_j = (x_j, y_j)$ de la siguiente de una secuencia ordenada temporalmente. La disparidad entre los puntos está dada por el vector de desplazamiento $d_{ij} = (x_i - x_j, y_i - y_j)$. El resultado de la correspondencia entre dos cuadros consecutivos es un conjunto de pares conjugados.

Para resolver el problema de correspondencia, se debe considerar: ¿Con qué criterio se afirma que dos puntos se corresponden? ¿Cuáles son las características que se comparan? ¿Qué limitaciones, si las hay, se imponen a los vectores de desplazamiento? Como parámetros para responder a estas preguntas, es indispensable tener presente las siguientes propiedades:
- *Similitud*: Es una medida de cuánto dos puntos se asemejan entre sí. Está dada por la similitud entre los objetos, determinada a partir de las características encontradas durante la segmentación.
- *Consistencia en el movimiento*: Debido a la inercia, el movimiento de una entidad física no puede cambiar instantáneamente. Si la secuencia es obtenida con una frecuencia tal que entre dos cuadros consecutivos no hay cambios drásticos, para la mayoría de los objetos las imágenes reflejan la suavidad de los movimientos.

Seguimiento de Trayectorias (Tracking)

Sea una secuencia de imágenes sobre las cuales se debe seguir una o más trayectorias. Si solamente hay un objeto, el problema del seguimiento puede resolverse de manera sencilla. En la presencia de muchas entidades moviéndose independientemente, se requiere el uso de restricciones basadas en la naturaleza de los objetos y sus movimientos. Teniendo en cuenta la propiedad de consistencia en el movimiento, se puede formular la *coherencia del camino*: implica que el movimiento de un objeto en cualquier punto en una secuencia de cuadros no va a cambiar abruptamente. Entonces la formulación de una solución al problema de correspondencia se ve simplificada puesto que se puede suponer que (a) la ubicación de un punto dado, (b) la velocidad escalar de un punto, y (c) la dirección de movimiento de un punto, permanecen relativamente sin cambios de un cuadro al siguiente.

La *función de desviación* o *función de trayectoria* se utiliza para evaluar las propiedades del movimiento en una secuencia. Su entrada es un trayecto y el valor de retorno debe ser inversamente proporcional a la suavidad de la trayectoria (o directamente proporcional al grado de desviación del camino).

Sea una trayectoria $T_i = \langle P_i^1, P_i^2, P_i^3, \cdots, P_i^n \rangle$, donde P_i^k representa un punto en la imagen *k*-ésima. Si las coordenadas de P_i^k están dadas por el vector X_i del cuadro, las coordenadas del *k*-ésimo cuadro pueden ser representadas como X_{ik}. Vectorialmente, $T_i = \langle X_{i1}, X_{i2}, X_{i3}, \cdots, X_{in} \rangle$.

La desviación d_i^k en el camino, del punto en el *k*-ésimo cuadro, está dada por $D_i^k = \phi\left(\overline{X_{ik-1}X_{ik}}, \overline{X_{ik}X_{ik+1}}\right)$, donde ϕ es una función de coherencia del camino (que da una idea de cuan coherente sería el movimiento de un objeto que pasa por estos puntos). La desviación para toda la trayectoria se define como $D_i = \sum_{k=2}^{n-1} d_i^k$.

Si hay *m* puntos en una secuencia de *n* cuadros, lo que resulta en *m* trayectorias, debería considerarse la desviación de todas las trayectorias, dada por $D(T_1, T_2, T_3, \cdots, T_m) = \sum_{i=k}^{m} \sum_{k=2}^{n-1} d_i^k$.

Luego, el problema de correspondencia se resuelve minimizando la desviación total D (o lo que es lo mismo, maximizando la suavidad del movimiento), para encontrar el conjunto de trayectorias correctas.

Si la frecuencia de muestreo de la cámara es suficientemente alta, el cambio en la dirección y velocidad de cualquier punto en movimiento en cuadros temporalmente consecutivos es suave. Esto se describe con la función de desviación $\phi\left(P_i^{k-1}, P_i^k, P_i^{k+1}\right) = \omega_1(1-\cos\theta) + \omega_2\left(1 - 2\frac{d_1 d_2}{d_1 + d_2}\right)$.

Vectorialmente: $\phi\left(P_i^{k-1}, P_i^k, P_i^{k+1}\right) = \omega_1\left(1 - \frac{\overline{X_{ik-1}X_{ik}X_{ik}X_{ik+1}}}{\left\|X_{ik-1}X_{ik}\right\|\left\|X_{ik}X_{ik+1}\right\|}\right) + \omega_2\left(1 - 2\frac{\sqrt{\left\|X_{ik-1}X_{ik}\right\|\left\|X_{ik}X_{ik+1}\right\|}}{\left\|X_{ik-1}X_{ik}\right\| + \left\|X_{ik}X_{ik+1}\right\|}\right)$.

El primer término es el producto vectorial de los vectores de desplazamiento y representa la *coherencia de dirección* (el producto vectorial es el ángulo entre los dos segmentos imaginarios entre los puntos k-1,k y k+1): cuanto menor es el ángulo, mayor es la desviación. El segundo término considera la media geométrica y aritmética de la magnitud, y representa la *coherencia de velocidad*. ω_1 y ω_2 son pesos seleccionados para asignar diferente importancia a los cambios en dirección y velocidad. Pueden elegirse en el rango de 0.00 a 1.00 tal que su suma sea 1.

Una de las principales dificultades con las secuencias es la oclusión, esto es, la desaparición total o parcial de objetos, o la aparición de otros. Los cambios en la información debido al movimiento y a las variaciones en la iluminación de la escena, pueden conducir a correspondencias incorrectas. Forzando las trayectorias para que satisfagan algunas restricciones locales y permitiendo que las mismas queden incompletas si es necesario, pueden obtenerse trayectorias aún en presencia de oclusión.

Una limitación del algoritmo de coherencia del camino es que asume que en todos los cuadros se dispone del mismo conjunto de puntos. Al buscar un conjunto de trayectorias se debe permitir obtener trayectorias incompletas que indiquen oclusión, aparición de objetos, o desaparición temporal de objetos por una mala calidad de la información. Más aún, deben imponerse ciertas restricciones sobre el máximo desplazamiento y la máxima desviación local posibles.

Dado un conjunto P^j de m_j puntos para cada uno de los n cuadros, puede encontrarse el máximo conjunto de trayectorias completas o parcialmente completas que minimiza la suma de desviaciones locales para todas las trayectorias obtenidas, sujeto a las condiciones de que la máxima desviación para cualquiera de las trayectorias no exceda $\phi_{\max}$ y el desplazamiento entre cualquier par de cuadros sucesivos para cualquier trayectoria sea siempre menor que $d_{\max}$. Para justificar los puntos faltantes debido a oclusión se usan *puntos fantasma* (puntos hipotéticos usados como relleno para extender todas las trayectorias sobre el conjunto de cuadros dado). Para que la noción de puntos fantasma sea útil, se deben definir los valores de desplazamiento y desviación local para una trayectoria que tiene tales puntos.

- Para computar el desplazamiento para T_i en el *k*-ésimo cuadro se define la función de desplazamiento:

$$Desp\left(P_i^k, P_i^{k+1}\right) = \begin{cases} Distancia\ Euclideana(P_i^k, P_i^{k+1}) & si\ ambos\ puntos\ son\ reales \\ d_{max} & en\ otro\ caso \end{cases}$$

Esto implica que un punto fantasma siempre se mueve en una cantidad fija $d_{\max}$.

- Para computar la desviación local para T_i en el *k*-ésimo cuadro se define la función de desviación:

$$Desv\left(P_i^{k-1}, P_i^k, P_i^{k+1}\right) = \begin{cases} 0 & si\ P_i^{k-1}\ es\ un\ punto\ fantasma \\ \phi(P_i^{k-1}, P_i^k, P_i^{k+1}) & si\ los\ tres\ puntos\ son\ reales \\ \phi_{max} & en\ otro\ caso \end{cases}$$

Esta definición de función de desviación local es equivalente a la función de coherencia del camino si los tres puntos son reales. Introduce una penalidad de ϕ_{max} en el caso de no tener un punto real para T_i en el cuadro corriente o el siguiente, y no penaliza si la trayectoria empieza en el cuadro que se está considerando.

El siguiente conjunto de pasos muestra a grandes rasgos el procedimiento de armado de las trayectorias.

Inicialización:

1. Para cada punto m_k en P^k, k=1, ..., n-1 (P^k conjunto de puntos del cuadro *k*), determinar el vecino más cercano en P^{k+1} dentro de la distancia $d_{\max}$. Resolver arbitrariamente en caso de múltiples alternativas.
2. Formar trayectorias iniciales encadenando los vecinos cercanos encontrados en cuadros sucesivos. Extender todas las trayectorias incompletas usando puntos fantasma para que vaya a lo largo de los *n* cuadros.
3. Para cada trayectoria del paso 2, formar una trayectoria adicional sólo con puntos fantasma

Ciclo de intercambio:

Para cada cuadro desde k = 2 hasta n-1
  Para i = 1 hasta m-1
    Para j = i+1 hasta m
      Si se cumplen las restricciones de d_{max}, calcular

$$G_{ij}^k = \left[\phi\left(P_i^{k-1}, P_i^k, P_i^{k+1}\right) + \phi\left(P_j^{k-1}, P_j^k, P_j^{k+1}\right)\right] - \left[\phi\left(P_i^{k-1}, P_i^k, P_j^{k+1}\right) + \phi\left(P_j^{k-1}, P_j^k, P_i^{k+1}\right)\right]$$

Tomar el par *ij* con máxima ganancia (G_{ij})
  Si la ganancia es mayor que cero entonces
    Intercambiar los puntos del cuadro *k+1*, P_i^{k+1} con P_j^{k+1}.

Terminación:

Repetir el ciclo de intercambio hasta que no haya más cuadros.

El número de trayectorias armadas en el paso 2 de la inicialización depende de la calidad de los datos. En el caso ideal, donde el mismo número *m* de puntos está consistentemente presente en todos los cuadros, sólo se formarán *m* trayectorias y ninguna tendrá puntos fantasmas. En el peor de los casos, cuando los puntos de alguno de los cuadros no tengan ninguna correlación con los puntos de cualquier otro, el número de trayectorias que se forme puede ser tan grande como la suma total de todos los puntos de todos los cuadros. En general, el número de trayectorias va a ser al menos m_{max} donde $m_{max} = \max(m_1, m_2, \ldots, m_n)$ y las diferentes trayectorias van a tener diferentes cantidades de puntos fantasma.

Se supone siempre un caso con varios objetos. En el caso particular en que se debe seguir a un objeto conocido, una solución para economizar tiempo de procesamiento es inspeccionar en cada imagen sólo la parte donde se estima que puede estar el objeto en función de su ubicación en los cuadros anteriores, y teniendo en cuenta el movimiento estimado del objeto. Por otro lado en este algoritmo de seguimiento de trayectorias se está viendo a los objetos como un conjunto de entidades indivisibles. Una situación más interesante es aquella en la que se conoce el modelo del objeto u objetos buscados, y a medida que se inspecciona la imagen, se pueden descartar los segmentos que no obedecen a ninguno de estos modelos.

Planteo del Problema y Solución Secuencial

Pensando el sistema como aplicable a la realidad, la fuente de datos debe provenir directamente de un dispositivo que capture las imágenes en tiempo real. A los fines de desarrollo y evaluación, los datos estuvieron disponibles completamente con anterioridad a la ejecución de la aplicación. Las imágenes fueron capturadas con una filmadora hogareña. El video analógico obtenido fue digitalizado y convertido en una secuencia de imágenes. Para generar un código donde la manipulación de imágenes sea sencilla y clara, se utilizaron imágenes en formato pgm, codificadas en la escala de grises de 0 a 255.

El siguiente algoritmo da una primera idea del ciclo principal de ejecución del sistema:

Mientras (hay cuadros)
    Tomar un cuadro
    Identificar los objetos del cuadro
Identificar y armar las trayectorias de los objetos hallados

¿Qué significa “mientras hay cuadros”? En el caso de imágenes provenientes de un dispositivo de captura en tiempo real, es equivalente a “mientras se estén captando imágenes”. Para imágenes disponibles con anterioridad, significa “mientras queden imágenes aún no procesadas”.

Dado que las imágenes de muestra fueron capturadas específicamente para este fin, no existió deterioro o mala calidad de las mismas. No obstante se notó que no mostraban una clara diferencia entre objetos y fondo (debido a la pobre iluminación en la captura), y por otra parte, que ante un movimiento acelerado de los objetos, algunos cuadros de la secuencia se veían borrosos. Esto originó la necesidad de realzar las imágenes para facilitar el reconocimiento. Esto se hizo calculando primero un promedio espacial para mejorar la definición, y luego destacando los puntos pertenecientes a objetos por medio de una función de ensanchamiento del contraste.

Para el reconocimiento de objetos se recorrieron las imágenes de izquierda a derecha y de arriba hacia abajo. Dado a que en la implementación el foco se centró en el seguimiento de trayectorias más que en el reconocimiento de objetos, se supuso que éstos eran bastante distinguibles del fondo, y que las características de interés eran mínimas: ubicación, área, intensidad promedio (nivel de gris o color). Estos atributos fueron suficientes para caracterizarlos, dado que no se buscaban objetos con determinadas características sino cualquier objeto de la escena. La técnica que se utilizó para la identificación de objetos dentro de una imagen combina el thresholdind con el labeling.

Una vez finalizado el recorrido de toda la imagen se depuró la información, descartando los segmentos tan pequeños o débiles en su color que no merezcan consideración, ya que simplemente pueden tratarse de manchas o sombras. Una restricción impuesta a las imágenes fue que los objetos no se tocaran. Al terminar el proceso se obtuvo como resultado una lista de los objetos encontrados en la escena con las características que los identifican.

La implementación es general en el sentido que se conoce que en la escena hay varios objetos, y que los mismos no son indivisibles uno del otro. Para casos más específicos, la solución puede mejorarse inspeccionando sólo la parte donde se estima que puede estar el objeto en función de su ubicación en los cuadros anteriores. Teniendo en cuenta que las imágenes provienen de la realidad, no tiene sentido buscar un objeto muy lejos de donde apareció en el cuadro precedente.

Correspondencia de Objetos y Seguimiento de las trayectorias

Se supone que la secuencia corresponde a una filmación donde la cámara es fija, y que el movimiento de los objetos es paralelo al plano de la imagen. Para el armado de las trayectorias, modificamos el algoritmo inicial (que suponía que toda la secuencia estaba disponible desde un principio) para el caso en que las imágenes se incorporan en tiempo real.

Mientras (hay cuadros)
Tomar un cuadro
Identificar los objetos del cuadro
Actualizar las trayectorias de los objetos hallados

Supongamos que el cuadro actual de la secuencia es el *k* y que en cada uno de los anteriores se identificaron P1, P2 . . .Pk-1 objetos. Hasta el momento se tienen armadas n trayectorias que van desde el cuadro 1 hasta k-1. El procedimiento debe asociar uno a uno los objetos del cuadro actual con cada una de las trayectorias que se están armando. El algoritmo que resuelve el problema es el siguiente:

Para cada objeto existente en el cuadro k+1 crear una trayectoria fantasma
// Calcula la máxima desviación global
Para i=1 hasta m-1
    Para j=i+1 hasta m
        Si en el intercambio se cumplen las restricciones de d_{max}*, calcular*
            $G_{ij}^{k} = \left[\phi\left(P_i^{k-1}, P_i^{k}, P_i^{k+1}\right) + \phi\left(P_j^{k-1}, P_j^{k}, P_j^{k+1}\right)\right] - \left[\phi\left(P_i^{k-1}, P_i^{k}, P_j^{k+1}\right) + \phi\left(P_j^{k-1}, P_j^{k}, P_i^{k+1}\right)\right]$
    Tomar el par ij *con máxima ganancia (*G_{ij}*)*

Si la ganancia G_{ij} *es mayor que cero entonces*
    Intercambiar los puntos del cuadro k+1, P_i^{k+1} *con* P_j^{k+1}.
Grabar el resultado de este paso en el archivo de trayectorias

Para implementar la función de desviación se deben tomar un par de decisiones:
- El método a utilizar para la distancia entre dos puntos. En este caso se usó *city-block*: $d = |i_1 - i_2| + |j_1 - j_2|$.
- Los pesos a emplear para la coherencia de dirección y la coherencia de velocidad en la fórmula, ω_1 y ω_2. En este caso no se dio preferencia a ninguna de las dos mediciones, por lo que $\omega_1 = \omega_2 = 0{,}5$

La información acerca de las trayectorias se mantiene en un archivo de trayectorias. Para simplificar el algoritmo de intercambio así como para ahorrar tiempo de cómputo, la información de trayectorias relacionada a los últimos tres cuadros se mantiene en memoria. El orden en que quedan grabados los objetos es el que resultó de la aplicación del algoritmo de seguimiento de trayectorias. En el caso de puntos fantasma como parte de una trayectoria, se guardan también con un valor de fila y columna inválida (-1,-1) que los distingue de un punto común.

La implementación permite además que la información plana (con formato de texto), que constituye las trayectorias encontradas, pueda ser representada gráficamente. Se genera una imagen que para cada objeto identificado, muestra con un trazo el recorrido que el mismo hizo a lo largo de la secuencia de imágenes. El siguiente algoritmo muestra el armado de las imágenes de trayectorias:

Sea MAX la cantidad de objetos en el primer cuadro.
Para cada uno de estos objetos generar una imagen asociada a su trayectoria.
Para cada cuadro
Para I=1 hasta MAX
    Si en el cuadro el punto existe
        Generar una línea desde el punto del objeto en la imagen anterior, hasta
            el punto del objeto en la imagen actual
    Si en el cuadro el punto no existe (fantasma)
        Proyectar la posición y dibujar en otro color o dejar en blanco
Para cada objeto nuevo que aparezca en el cuadro
    Incrementar MAX en 1
    Crear una nueva imagen de trayectoria.

Solución Paralela

La paralelización se realizó sobre una red de computadoras heterogéneas y soporte MPI [GDB95] [Mil98] [Mor94] [MPI1] [MPI2] [MPI3] [PBM] [Sim97] [Sni96] [Ste96] [Wel96].La primera operación tratada fue el realce del contraste mediante un promedio espacial. Para el código paralelo, un proceso root o master distribuye por columnas la imagen entre todos los procesos y recopila los resultados armando la imagen final.

El proceso de reconocimiento de objetos combina thresholding con labeling. La paralelización se basa en la misma idea pero introduce algunos detalles. El root reparte las columnas en forma equitativa entre los procesos y realiza una parte de la operación general. Cada proceso recorre la subimagen extrayendo la información de los objetos candidatos. Luego cada uno envía la información al raíz de modo que el resultado

del proceso de reconocimiento de objetos en la imagen queda centralizado en un único nodo. Para resolver el caso de objetos que pertenecen a más de una subimagen, cada proceso identifica los candidatos "pegados" al borde izquierdo de su subimagen, y luego “se olvida” que encontró ese objeto; se lo delega al proceso de la izquierda y lo descarta de su propio conjunto de objetos. Al recibir la información del proceso que tiene como vecino a la derecha, verifica si coincide con la frontera de alguno de los objetos lindantes con el borde derecho. De ser así, el proceso unifica la información, sumando la del objeto recibido con la del objeto propio. Finalmente, cada proceso envía al maestro la información de los objetos encontrados, con lo cual termina el procedimiento.

Una vez identificados los objetos de las imágenes, deben armarse las trayectorias. Recordemos que en la solución secuencial, siendo P_1, P_2, ..., P_k los conjuntos de objetos encontrados en una secuencia de *k* imágenes, para cada imagen P_i se tomaban los objetos y se los “enganchaba” en las trayectorias armadas para las imágenes anteriores siguiendo el algoritmo. La información relacionada con las distintas trayectorias guarda en un archivo a medida que se avanza en el procedimiento. En memoria principal siempre se tiene la información de los últimos tres cuadros por performance y facilidad de procesamiento. Repasando la solución dada, el procedimiento secuencial para realizar el seguimiento de trayectorias de objetos se puede resumir de la siguiente manera: es un ciclo repetitivo en el cual, a medida que se dispone de nuevos cuadros de la secuencia, se actualizan las trayectorias armadas para los cuadros anteriores, con los objetos identificados en el nuevo cuadro. Es decir, sea la instancia donde hay que procesar el cuadro *i*. Se tienen N trayectorias incompletas, identificadas en los *i-1* cuadros anteriores. El siguiente paso es, entonces, "enganchar" en las trayectorias anteriores los *x* objetos identificados en el cuadro *i*. Para hacer esto, es necesario realizar varias operaciones de comparación para determinar cuál es la disposición de objetos que da un mejor conjunto de trayectorias.

Como resultado de estas operaciones probablemente haya que intercambiar el orden de algunos de los objetos hallados. Si es así, el ciclo se repite hasta que no haya más intercambios. Si *n* es el máximo entre la cantidad de trayectorias armadas, y el número de objetos encontrados en el nuevo cuadro, la cantidad de operaciones de comparación o verificación a realizar en cada ciclo es *(n-1) + (n-2) + ... + 1*. Puede comprobarse que esta suma es igual a $\sum_{i=1}^{n-1}\left[(n-1)^2 + n - 1\right]/2$:

Recuérdese que cada operación a que se hace mención es una función que dadas dos ternas de puntos u objetos, donde cada uno corresponde a una de tres imágenes consecutivas, determina una posible trayectoria verificando qué combinación da una menor desviación total (si las dos trayectorias dadas, o las que resultan de intercambiar el tercer punto de ambas ternas). Cuando el resultado es favorable para las ternas con el punto intercambiado es cuando se debe realizar el intercambio mencionado. Por supuesto la cantidad de intercambios y de ciclos necesarios dependen del conjunto de datos de muestra.

Una paralelización consiste en repartir equitativamente esa cantidad de operaciones entre los procesos. Si el número de operaciones no fuera múltiplo exacto del número de procesos, las sobrantes se repartirían una para cada uno de los primeros procesos. Por lo demás, todos realizarían las mismas funciones, con una copia local de los datos de las imágenes más recientes. Los siguientes son algunos ejemplos de la cantidad de operaciones según el número de procesos: Para un número pequeño de objetos el reparto de las operaciones resultaría en una cantidad minúscula de procesamiento a realizar por cada uno, sumado a los tiempos de comunicación. Ante esto lo mejor es descartar esta posibilidad y buscar una alternativa más adecuada.

Para ello, miramos la solución desde un nivel de menos refinamiento. El problema en su expresión general consiste, en identificar los objetos en una secuencia para luego seguir sus trayectorias. La actualización de las trayectorias puede hacerse siempre que se haya completado la identificación de objetos en al menos tres imágenes de la secuencia. Al disponer de más de un proceso, en lugar de paralelizar el algoritmo interno del seguimiento de trayectorias, separamos las dos tareas centrales entre los distintos procesos. Esto es, por un lado se tiene un conjunto de procesos que realizan la identificación, mientras que un proceso aparte se ocupa de actualizar las distintas trayectorias. Con esto se logra superponer la actualización de trayectorias con la fase de identificación de objetos en la imagen siguiente. Si se viera que ese tiempo es muy pequeño y mantiene al

proceso responsable ocioso por mucho tiempo, se podría decidir que esta parte del procesamiento la realice uno de los procesos del primer grupo.

Evaluación de los Resultados

Uno de los objetivos originales fue obtener resultados aceptables para problemas en tiempo real. Se analizó speedup y eficiencia de la solución paralela utilizando distinta cantidad de procesadores y procesos, y diferentes tamaños de imágenes.

Idealmente se intenta que la eficiencia tienda a 1. Si se toma el ideal como parámetro, se puede medir cuán lejos está la implementación de ese ideal. En el mejor caso, el tiempo para ejecutar una solución decrece proporcionalmente al número de procesos. Esto es, si N = número de procesos y T_N = tiempo para usar el problema usando N procesos, entonces a medida que *N* crece, T_N debería decrecer. En el caso ideal:

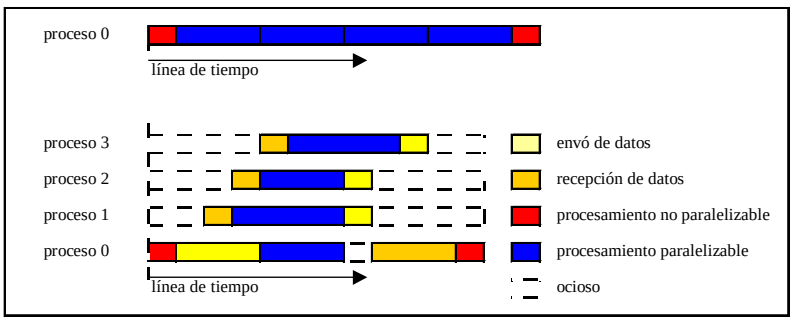
$T_N = T_1 / N$ o $N = T_1 / T_N$. Esta última fórmula da un indicador de *speedup*. En general $T_1 / T_N \leq N$. El cociente entre speedup real en N procesos y el ideal, da una medida para la *eficiencia*. Formalmente: *eficiencia = real / ideal = speedup real en N procesos / N.* Valores bajos de eficiencia representan desperdicio de potencia de procesamiento [Hwa93] [Kum94][Lei92].

En general para cada parte de la solución paralela el tiempo total de ejecución se puede dividir en dos componentes. Por un lado la parte del problema que fue paralelizada, donde agregar procesadores realmente reduce los tiempos, incluso muy cerca del ideal. Sea $\boldsymbol{T_p}$ el tiempo ocupado para ejecutar esta parte del problema en la solución secuencial. Por otro lado está la parte del problema inherentemente secuencial o no paralelizable, que no va a ser más rápida agregando procesadores. Llamando $\boldsymbol{T_S}$ al tiempo que lleva ejecutar esta parte del problema, $T_1 = T_S + T_p$

T_S es un límite inferior para el tiempo de procesamiento, por lo tanto en el mejor de los casos $T_N = T_S + (T_p / N)$. La fracción del tiempo total de ejecución correspondiente a la parte no paralelizable, T_S / T_1, impone un límite en la reducción de los tiempos independientemente del número de procesos. Cuanto más grande sea esta fracción, menos eficiente será la solución paralela. Es importante tener en cuenta que este factor no es una constante, sino que varía con el tamaño del problema: a medida que éste cambia, el factor T_S / T_1 también. Típicamente T_p crece mucho más rápido que T_S. Como consecuencia, una completa evaluación debe extenderse al menos sobre dos parámetros: el tamaño del problema y el número de procesadores. A esto hay que sumar que la performance puede ser seriamente afectada tanto por operaciones de entrada/salida, como por la cantidad de comunicaciones entre procesos.

Para poder determinar las características que debían tener las secuencias de imágenes de modo que la evaluación contemplara todas las situaciones posibles, se revisaron en primer lugar los algoritmos de cada parte del sistema. A partir de allí se definieron distintos casos de prueba y se generaron las secuencias de imágenes necesarias. Luego se evaluaron los algoritmos sobre cada uno de estos casos variando la cantidad de procesos y de máquinas involucrados.

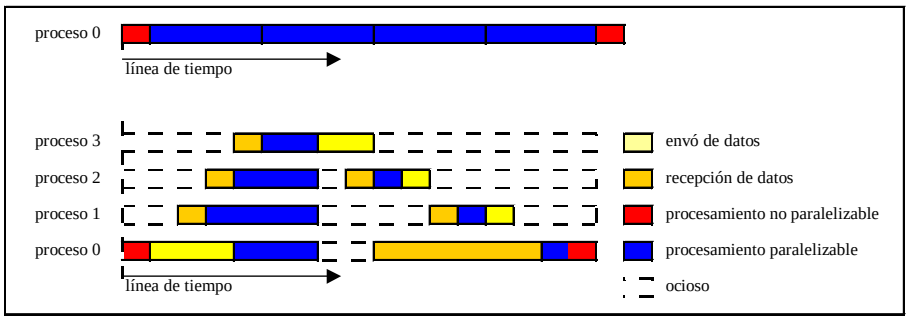
En el realce, T_S incluye la obtención de la imagen, apertura y cierre de archivos y reconocimiento del formato (por ejemplo, averiguar sus dimensiones). T_p incluye calcular el valor para cada punto de la nueva imagen. En el caso de más de un proceso, este tiempo no es puramente tiempo de procesamiento, sino que lleva consigo los tiempos de envío y recepción de datos entre los procesos. Con un gráfico temporal:



El tamaño del problema en este caso es directamente proporcional al tamaño de la imagen que se está procesando. Si se tiene en cuenta que cada proceso recibe además de las columnas de datos a procesar, las dos columnas adyacentes a éstas, puede verse que cuantos más procesos haya, más tráfico de datos se genera. Pero al mismo tiempo, el agregado de procesos supone una disminución en tiempo de procesamiento por proceso. Por lo tanto es necesario buscar un equilibrio entre estos dos puntos.

Para medir la performance de la solución para el realce se generaron casos de prueba con diferentes tamaños de imagen (tamaño del problema) y distinto número de procesos. Los resultados mostraron que, independientemente del tipo de imágenes, los tiempos que consume la tarea del realce son similares. Lo primero que se observó es que una secuencia de imágenes de tamaño chico no es la mejor candidata para ser realzada en paralelo. En este caso los tiempos que le tomó a un solo proceso realizar la operación fueron mejores que los que resultan de la parelelización. Por otro lado, para un tamaño mayor de imagen, el speedup fue de un 22% sobre el tiempo base, al incorporar un segundo proceso; no es un resultado óptimo, pero resulta positivo. No ocurrió lo mismo agregando un tercer proceso, con el cual los tiempos de procesamiento se incrementaron nuevamente.

En segundo término se evaluó el proceso de reconocimiento. Aquí, T_S incluye la obtención de la imagen, y el reconocimiento del formato, y T_p es el tiempo que toma identificar los objetos, contemplando en el caso de la solución paralela que este tiempo lleva consigo el envío de datos entre procesos, más el eventual "ensamblado" de objetos que hayan quedado divididos en el reparto. En un gráfico temporal:



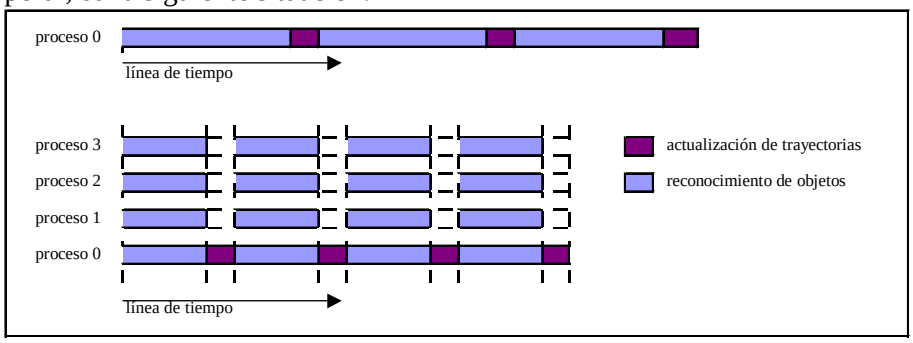
El tamaño del problema en este caso depende de dos factores principales:
- el tamaño de la imagen. Cuantas más filas y columnas haya, tantos puntos o pixels más hay que inspeccionar para encontrar la información de los objetos.
- el "juego de datos" particular. La cantidad, tamaño y disposición de los objetos influyen directamente en el tiempo de procesamiento, y en la cantidad de información que tendrá que "viajar" entre los procesos.

Por ejemplo, un problema grande estará caracterizado por imágenes de alta resolución, o con muchos objetos, o con objetos cuya forma sea alargada de manera que al dividir la imagen los mismos queden "partidos". Con respecto a la cantidad de procesos, al igual que en el caso anterior, al repartirse dos columnas más por proceso, el tráfico aumenta con el número de procesos, pero cada uno de ellos tiene menos datos a inspeccionar. La desventaja es que al estar más partida la imagen, existe la posibilidad de partir más los objetos, lo que agrega tiempo de "ensamblado" de objetos al final.

Para medir la performance de las soluciones propuestas para el reconocimiento se generaron casos de prueba con imágenes que resultan en distintos tamaños de problema, y con distintas cantidades de procesos. Las distintas pruebas fueron agrupadas según las características mencionadas, de modo que la evaluación resulte más interesante. Así se tuvieron resultados para pruebas agrupadas por cantidad de objetos y para pruebas agrupadas por tamaño de los mismos. No se consideró como criterio de agrupamiento la disposición de los objetos puesto que al tratarse de secuencias donde los objetos se mueven, dentro de una misma prueba esta característica no es fija; los objetos están en un momento en un sector que puede ser muy distinto del lugar donde se encuentran al final de la secuencia.

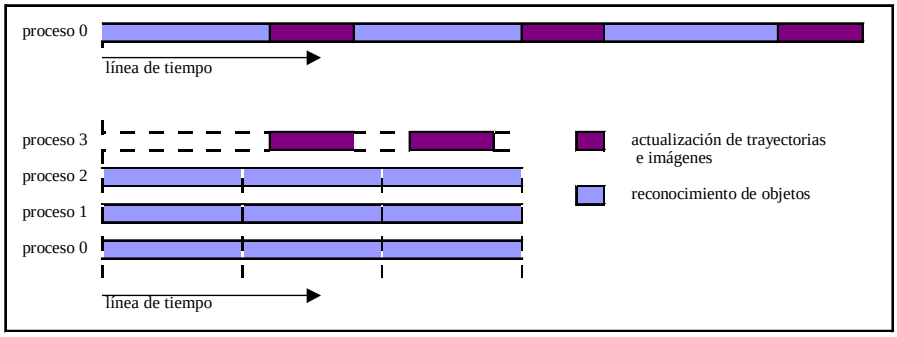
Las pruebas con respecto al algoritmo de reconocimiento revelan mejores resultados que en el caso anterior, aunque siempre en el caso de las imágenes de mayor tamaño. Par secuencias de imágenes chicas, en la mayor parte de los casos los tiempos aumentan con la cantidad de procesos (excepto cuando las imágenes contienen objetos grandes, pero el beneficio es muy poco y la eficiencia se reduce tanto que el resultado final resulta pobre). Para el caso de las imágenes más grandes en cambio, la incorporación de procesos mejora los tiempos, principalmente en los casos en que las imágenes contienen objetos grandes. Además el speedup que se logra al incorporar un tercer proceso es mejor aún que el que se obtiene al pasar de uno a dos procesos.

Finalmente resta evaluar el proceso de seguimiento de trayectorias de objetos. La solución dada, representada en forma temporal, da la siguiente situación:



Esto se asemeja al caso de reconocimiento. La diferencia la marca el procesamiento que debe realizar el maestro o raíz luego de analizada cada imagen. Por esto, para responder acerca de la dimensión del problema son válidos los mismos conceptos planteados para aquel caso: cantidad de objetos, disposición, etc. Además, dado que la cantidad de operaciones a realizar (en cuanto a la actualización de las trayectorias) depende de las características de los movimientos de los objetos, la velocidad y la disparidad con que éstos se muevan influirán también en el tiempo de ejecución total de las soluciones tanto secuencial como paralela.

La tarea puntual de actualización de trayectorias, como ya se evaluó, es una fracción muy chica de todas las tareas involucradas. Por esta causa, no tendría sentido dedicar un proceso exclusivamente para que la realice. Sí podría hacerse en el caso planteado como mejora: generar imágenes que muestren gráficamente la trayectoria que van dibujando los objetos en movimiento. En ese caso un proceso aparte actualizaría las trayectorias y también generaría estas imágenes, obteniendo el siguiente gráfico:



Para medir la performance del proceso de seguimiento de trayectorias se generaron secuencias de imágenes que varían la dimensión del problema: secuencias con dos o tres objetos, secuencias con más de tres objetos, secuencias con objetos con movimientos lentos y parejos, secuencias con objetos con movimientos rápidos y dispares.

De la misma manera que en el caso del reconocimiento, las pruebas se agruparon según las características mencionadas. Por un lado, pruebas agrupadas por la velocidad y coherencia de los movimientos (movimiento lento y parejo, velocidad moderada y pareja, velocidad alta y movimiento parejo, velocidad y coherencia dispares), y por otro pruebas agrupadas por cantidad de objetos

Es importante tener en cuenta que las pruebas del algoritmo de seguimiento de trayectorias se refieren a la prueba integral que va desde el realce de cada cuadro individual hasta el resultado final. No se trata de la prueba del algoritmo exclusivo del seguimiento

Estas pruebas abarcan todo el ciclo del procesamiento de las imágenes. Como la porción exclusiva del mantenimiento de las trayectorias es una parte muy pequeña del algoritmo, y además es tarea de un único proceso, es de esperar que el resultado no se revierta respecto de las pruebas anteriores sobre el realce y el reconocimiento de objetos sobre las imágenes. Esto fue de hecho lo que ocurrió.

En resumen:

- Para imágenes pequeñas (en particular 320 x 240) no resulta adecuado paralelizar los algoritmos, ya que en proporción resulta mayor el tiempo de comunicación que el que cada uno ocupa en realizar las distintas tareas. El trabajar con más de un proceso solamente bajó los tiempos cuando los objetos de las imágenes eran grandes, pero el speedup resultó tan bajo que ni siquiera en estos casos merece tenerse en cuenta.

- Para imágenes más grandes (en particular 720 x 540) los resultados fueron aceptables no tanto en el algoritmo inicial de realce de imágenes, sino en particular en el de reconocimiento de objetos y por supuesto en la prueba conjunta de todo el sistema. En especial, los resultados fueron mejores en los casos en que las imágenes contenían objetos de tamaño grande en relación al tamaño del cuadro. Por último, al momento de evaluar los algoritmos según los tipos de movimiento de los objetos, se observa en los casos donde los mismos son más lentos y parejos, los resultados son mejores aunque no por una gran diferencia respecto de los demás.

Conclusiones

Luego de haber completado la implementación y las evaluaciones de los algoritmos descriptos, el resultado obtenido cumplió con las expectativas planteadas, a saber:

- Se desarrolló por completo una solución para identificar objetos en una secuencia de imágenes e identificar las trayectorias de sus movimientos.

- Se identificaron los casos en que la solución paralela resulta buena, y aquellos en los que es preferible adoptar la solución tradicional.

Por otra parte, a lo largo del proyecto se plantearon mejoras o alternativas a la solución implementada. Las mismas se resumen aquí junto con algunas sugerencias adicionales:

a) *Con respecto al proyecto*

- Generar imágenes adicionales que reflejen el movimiento de los objetos de la escena.

- Identificar objetos basados en características que los distingan unos de otros. En el presente trabajo, los objetos eran indivisibles. Esta mejora se complementa con una modificación en el algoritmo de seguimiento que aparee los objetos no solo por su posición en los cuadros consecutivos sino también por las características propias que los individualizan.

- En el caso en que se tenga que seguir solamente a un objeto, modificar el algoritmo de reconocimiento de manera de inspeccionar solo un sector de la imagen dependiendo de dónde se haya encontrado el objeto en los cuadros anteriores. Esto mejora la solución, sin desperdiciar tiempo de procesamiento.

- Modificar o mejorar los algoritmos planteados para responder mejor según las distintas secuencias de imágenes a evaluar. El proyecto está implementado de manera tal que resulta simple cambiar cualquiera de sus partes por otra que responda mejor a los requerimientos (por ejemplo, se puede querer cambiar el algoritmo de reconocimiento de imágenes, o trabajar con otro formato de imágenes).

b) *Con respecto a la performance*

- Evaluar cada cuánto tiempo es necesario realizar la selección del threshold para la segmentación. Dada la natural suavidad en los movimientos de las imágenes, no es necesario recalcular este valor para cada cuadro

ya que no se obtendrían cambios significativos de una imagen a la siguiente. Sin embargo, la frecuencia necesaria puede depender de la velocidad de los movimientos de los objetos.

- Implementar una alternativa para manejar la información auxiliar (labels y objetos) en memoria en lugar de archivos. Esta opción sería mejor en tiempo, pero requeriría disponer de máquinas provistas de memoria suficiente.

- Evaluar la posibilidad de saltear cuadros en el caso en que se conozca de antemano que los objetos en la escena se mueven muy lentamente, lo que permitiría obtener la misma información con mejores tiempos.

- Para el caso que se conozca que los objetos son de tamaño considerable, implementar una variante que no inspeccione cada pixel sino que tome uno de cada dos o más, de modo de reducir el procesamiento. Es esperable que los resultados no cambien, pero probablemente sí los tiempos.

- Variar la distribución de las tareas entre los distintos procesos. Por ejemplo, que el realce lo realicen dos procesos, mientras que la tarea de reconocimiento la realicen entre tres. Esta distribución tiene que ver con aprovechar las conclusiones obtenidas de las pruebas realizadas. Por otro lado si se dispone de más máquinas, otra posibilidad es distribuir la tarea de modo que un grupo realice el realce, y otro grupo de procesos en las otras máquinas realice el reconocimiento.

Además, puede migrarse la implementación hacia un multiprocesador homogéneo (por ejemplo, el hipercubo de transputers disponible en el Laboratorio de Procesamiento Paralelo de la Facultad de Informática UNLP), con el objetivo de comparar la performance sobre ambas plataformas.

Bibliografía

[Akl97] Akl S, “Parallel Computation. Models and Methods”, Prentice-Hall, Inc., 1997.
[Bax94] G. A. Baxes, "Digital Image Processing. Principles and Applications", John Wiley & Sons Inc., 1994.
[Bri95] Brinch Hansen, P., “Studies in computational science: Parallel Programming Paradigms”, Prentice-Hall, Inc., 1995.
[Gon92] R. C. González, R. E. Woods, "Digital Image Processing", Addison-Wesley Publishing Comp., 1992.
[GDB95], GDB/RBD, "MPI Primer / Developing with LAM", The Ohio State University, 1995
[Gup93] Gupta A., Kumar V., “Performance properties of large scale parallel systems”, Journal of Parallel and Distributed Computing, November 1993.
[Har92] R. M. Haralick, L. G. Shapiro, “Computer and Robot Vision”, Addison-Wesley Publishing Company, 1992.
[Hwa93] K. Hwang, "Advanced Computer Architecture. Parallelism, Scalability, Programmability", McGraw Hill, 1993.
[Jai89] A. Jain, "Fundamentals of Digital Image Processing", Prentice Hall Inc., 1989.
[Jai95] R. Jain, R. Kasturi, B. G. Schunck, “Machine Vision”, McGraw-Hill International Editions, 1995.
[Kum94] Kumar V., Grama A., Gupta A., Karypis G., “Introduction to Parallel Computing. Desing and Analysis of Algorithms”, Benjamin/Cummings, 1994.
[Law92] H. Lawson, "Parallel processing in industrial real time applications", Prentice Hall 1992.
[Lei92] F. T. Leighton, “Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes”, Morgan Kaufmann Publishers, 1992.
[Mil98] Miller R., Stout Q. F., “Algorithmic Techniques for Networks of Processors”, CRC Handbook of Algorithms and Theory of Computation, M. J. Atallah, ed, 1998.
[Mor94] H. S. Morse, “Practical Parallel Computing”, AP Professional, 1994.
[MPI1] http://www.mcs.anl.gov/pub/mpi. Información sobre implementaciones y sobre MPI en general - Argonne National Laboratory
[MPI2] http://www.erc.msstate.edu/mpi. Información sobre implementaciones y sobre MPI en general – Mississippi State University.
[MPI3] ftp://info.mcs.anl.gov/pub/mpi. Implementación MPICH de Argonne National Laboratory y Mississippi State University.
[PBM] www.acme.com/software/pbmplus Image file format conversion package.

[Sch92] R. Schalkoff, “Pattern Recognition. Stadistical, Structural and Neural Approches”, 1992
[Sim97] Sima D, Fountain T, Kacsuk P, “Advanced Computer Architectures. A Design Space Approach”, Addison Wesley Longman Limited, 1997.
[Sni96] Snir M., Otto S., Huss-Lederman S., Walker D., Dongarra J., "MPI: The Complete Reference", The MIT Press, 1996
[Ste96] Steenkiste P., “Network-Based Multicomputers: A Practical Supercomputer Architecture”, IEEE Transactions on Parallel and Distributed Systems, Vol. 7, No. 8, August 1996, pp. 861-875
[Tin98] Tinetti F., De Giusti A., "Procesamiento Paralelo. Conceptos de Arquitectura y Algoritmos", Editorial Exacta, 1998.
[Wel96] Welsh M., Kaufman L., "Running LINUX", O’Reilly & Associates, Inc, 1996 (Second Edition)
[Zom96] Zomaya A., “Parallel Computing. Paradigms and Applications”, Int. Thomson Computer Press, 1996.

.